Comp. & Maths. with Appls., Vol. 2, pp. 225–232. Pergamon Press 1976. Printed in Great Britain

SUBSPACE SELECTION ALGORITHMS TO BE USED WITH THE NONLINEAR PROJECTION METHODS IN SOLVING SYSTEMS OF NONLINEAR EQUATIONS*

D. D. GEORG

Computer Science Department, Colorado State University, Fort Collins, Colorado 80523, U.S.A.

and

R. F. KELLER

Ames Laboratory, ERDA, Mathematics Department, Computer Science Department, Iowa State University, Ames, Iowa 50011, U.S.A.

Communicated by E. Y. Rodin

(*Received March* 1976, *and in revised form June* 1976)

Abstract—The nonlinear projection methods are minimization procedures for solving systems of nonlinear equations. They permit reevaluation of n_k, $1 \le n_k \le n$, components of the approximate solution vector at each iteration step where n is the dimension of the system. At iteration step k, the reduction in the norm of the residue vector depends upon the n_k components which are reevaluated. These n_k components are obtained by solving a linear system.

We present two algorithms for determining the components to be modified at each iteration of the nonlinear projection method and compare the use of these algorithms to Newton's method. The computational examples demonstrate that Newton's method, which reevaluates all components of the approximate solution vector at each iteration, can be accelerated by using the projection techniques.

1. INTRODUCTION

In the past twenty years, several new methods for solving simultaneous nonlinear equations have been introduced and many modifications have been suggested for Newton-type methods, [2, 3, 7, 9, 13, 19, 21, 23, 39, 43]. The equation to be solved is

$$f(x) = 0 \tag{1.1}$$

where x is an n-column vector and f is a column vector of n functions f_j, $j = 1, 2, \ldots, n$. If x^i is the ith approximation to the solution of (1.1), then Newton's method is defined by

$$x^{i+1} = x^i - J(x^i)^{-1} f(x^i) \tag{1.2}$$

where $J(x^i)$ is the Jacobian matrix evaluated at x^i.

Computational accelerations to Newton's method have been obtained by approximating $J(x^i)$ and reducing the computations required to evaluate (1.2). Projection methods used in conjunction with the algorithms given in this paper determine another technique for reducing the computations required by Newton-type methods.

Nonlinear systems of equations are solved with projection methods by minimization in such a way that at each iteration an arbitrary number of components of the approximate solution vector can be modified. Earlier studies have established convergence criteria [30] and have established the conditions under which Newton's method is equivalent to a projection method [22]. We present two algorithms for choosing the components to be modified by the projection method. Both algorithms take advantage of the magnitudes of the angles between the residue vector and the columns of $J(x^i)$.

*Work performed for the U.S. Energy Research and Development Administration under Contract No. W-7405-eng-82. Work partially supported by a C.S.U. Faculty Research Grant.

II. NONLINEAR PROJECTION METHODS

If n_k components of the approximate solution vector are modified at step k, step k is classified as an n_k dimensional projection. The indices of the modified components will be denoted by $B_k = \{i_1, i_2, \ldots, i_{n_k}\}$. The change components, dx_i^k, $i \in B_k$, are obtained at step k by solving the n_k linear equations

$$\sum_{i \in B_k} (J_i^k, J_r^k)\, dx_i^k = -(f(x^k), J_r^k), \quad r \in B_k \tag{2.1}$$

where J_i^k is the i^{th} column of the Jacobian evaluated at x^k and (a, b) denotes the inner product of vectors a and b on the Euclidean space. If $B_k = \{1, 2, \ldots, n\}$ for all k then this is simply the Gauss–Newton method [31, pp. 267–271].

Let r^k denote the residue vector at the k^{th} step. By definition,

$$r^k = f(x^k)$$

If the components of x^k prescribed by B_k are modified at step k, then r^{k+1} will be perpendicular to the n_k columns $i_1, i_2, \ldots, i_{n_k}$ of $J(x^k)$. The algorithms presented in this paper utilize this result in selecting B_k, $k = 1, 2, \ldots$.

Let δr^k denote the difference between the Euclidean norms of the two successive residue vectors r^k and r^{k+1}. The simple result that δr^k can be bounded as a function of B_k is shown as follows. By the mean value theorem, one has

$$f_j(x^{k+1}) = f_j(x^k) + (S_k{}^T d_k, J_j(x^k)) + (S_k{}^T d_k, (f_j''(z), S_k{}^T d_k))$$

where $z \in (x^k, x^{k+1})$, $j = 1, 2, \ldots, n$ and S_k denotes an $n_k \times n$ matrix comprised of those rows of the n-dimensional identity matrix which correspond to the indices of the columns of the Jacobian matrix which are contained in B_k, d_k denotes the n_k component vector of change components that are solved for at the k^{th} step, $J_j(x^k)$ denotes the j^{th} row of the Jacobian evaluated at x^k, and $f_j''(z)$ denotes the Hessian of $f_j(z)$.

δr^k can be expanded using (2.2):

$$\begin{aligned}\delta r^k &= \|f(x^k)\| - \|f_1(x^k) + (S_k{}^T d_k, J_1(x^k)) + (S_k{}^T d_k, (f_1''(z_1), S_k{}^T d_k)), f_2(x^k) \\ &\quad + (S_k{}^T d_k, J_2(x^k)) + (S_k{}^T d_k, (f_2''(z_2), S_k{}^T)), \ldots, f_n(x^k) \\ &\quad + (S_k{}^T d_k, J_n(x^k)) + (S_k{}^T d_k, (f_n''(z_n), S_k{}^T d_k))\| \\ &= \|f(x^k)\| - \|f(x^k) + J(x^k) S_k{}^T d_k - \theta_k\|\end{aligned} \tag{2.3}$$

where

$$z_i \in (x^k, x^{k+1}), \quad i = 1, 2, \ldots, n$$

$$\theta_k = -\begin{bmatrix} (S_k{}^T d_k, (f_1''(z_1), S_k{}^T d_k)) \\ (S_k{}^T d_k, (f_2''(z_2), S_k{}^T d_k)) \\ \vdots \\ (S_k{}^T d_k, (f_n''(z_n), S_k{}^T d_k)) \end{bmatrix}$$

Now, using the triangle inequality, a bound is obtained directly:

$$\delta r^k \le \|\theta_k - J_k S_k{}^T d_k\| \tag{2.4}$$

The implication of (2.4) is that the bound on δr^k, the difference between the Euclidean norms of the two successive residue vectors r^k and r^{k+1}, is dependent on the set of components which are modified at step k. This bound is useful in motivating the consideration of the effects of different sequences $\{B_k\}_{k=0}^{\infty}$ on the problem solution. To illustrate this effect, Table 1 contains values of $\|r^1\|$ that are computed for alternative single projections on Example 1.

Example 1

$$f_1(x) = x_1x_2/x_3 - x_4x_5$$

$$f_2(x) = x_1x_3/x_2 + x_4 + x_5 - 2.0$$

$$f_3(x) = x_2x_3/x_1 - x_4^2$$

$$f_4(x) = x_1/(x_2x_3) - x_5^2$$

$$f_5(x) = x_2/(x_1x_3) - x_4^2 + x_5^2 - 1.0$$

$$x^0 = (1, 1, 1, -2, 2)$$

$$r^0 = (5, -5, -3, -3, 0)$$

$$\|r^0\| = 8.24621$$

Table 1. Results of single projections on Example 1

n_0	B_0	$\|r^1\|$
5	(1, 2, 3, 4, 5)	7.71497
4	(1, 2, 3, 4)	7.80230
4	(1, 2, 3, 5)	8.90330
4	(1, 2, 4, 5)	6.20509
4	(1, 3, 4, 5)	2.35483
4	(2, 3, 4, 5)	7.71499
3	(1, 3, 5)	11.69840
3	(1, 4, 5)	2.82843
3	(3, 4, 5)	1.51791

In Table 1, the reduction in the norm of the residue vector that was exhibited by the projection method for $n_0 = 5$ is the same reduction that would be obtained with one step of Newton's method. When $n_0 \neq 5$, it is possible to obtain either a smaller or larger value of δr^0 compared to δr^0 with $n_0 = 5$. The goals of the algorithms presented in this paper are to select a subspace B_k at each step k which will result in a relatively large value of δr^k.

A particular projection method is determined by the sequence of changes $\{B_k\}_{k=0}^{\infty}$ which are used to solve a given problem. In the next two sections of this paper, two algorithms are described for choosing B_k dynamically at each iteration step k.

III. ANGLE ALGORITHM

As was previously mentioned, the result of an n_k dimensional projection is that the residue vector, r^{k+1}, is perpendicular to the n_k columns of the Jacobian matrix evaluated at x^k that determine the projection subspace. Thus, if the angle between r^{k+1} and the i^{th} column of the Jacobian evaluated at x^k, denoted by $\theta_{r^{k+1}J_i^k}$ is approximately 90°, then choosing column i alone as a subspace would not produce a good reduction in the norm of the residue vector. The inclusion of such a column i in a larger projection subspace may or may not be beneficial to the norm reduction. However, numerical results support the contention that it is usually not beneficial to include the columns which are almost perpendicular to the residue vector in the projection subspace.

Since working with the angles directly is not convenient, the angle algorithm utilizes the cosines of the angles squared. By definition,

$$\cos^2 \theta_{r^kJ_i^k} = \frac{(r^k, J_i^k)^2}{(r^k, r^k)(J_i^k, J_i^k)}. \tag{3.1}$$

The angle algorithm described below computes the values of $\gamma_i = \cos^2 \theta_{r^kJ_i^k}$, $i = 1, 2, \ldots, n$ at each iteration.

Angle algorithm

A1. Select a tolerance T

A2. $B_k \leftarrow \phi$

A3. $i \leftarrow 1$
A4. Compute $\gamma_i = \cos^2 \theta_{r^k J_i^k}$
A5. Compare γ_i and T.
If $\gamma_i > T$, $B_k \leftarrow B_k \cup \{i\}$
A6. $i \leftarrow i + 1$
A7. Repeat A4, A5, A6 until $i > n$
A8. If $B_k = \phi$ set $B_k = \{1, 2, \ldots, n\}$

The angle algorithm excludes column i from B_k whenever $\gamma_i \leq T$. For example, if $T = 0.25$, then no column would be included in the projection subspace if it were within 30° of being perpendicular to the residue vector. The value of T which is used will affect the B_k which are computed. Computationally, it has been observed that better results are obtained whenever $0.1 < T < 0.4$.

IV. φ ALGORITHM

The φ algorithm utilizes an alternative form of computing the γ_i's as a basis for determining projection subspaces. This algorithm is motivated by the observation that if column i was constained in the projected subspace in order to obtain x^{k+1}, then $\theta_{r^{k+1} J_i^k} = 90°$. If this were the case, column i should be, using the same reasoning motivating the angle algorithm, excluded from the next projection subspace.

The alternative form of computing γ_i is obtained by expressing $\cos^2 \theta_{r^{k+1} J_i^{k+1}}$ in terms of r^{k+1}, J_i^k and Δ_i^k where

$$\Delta_i^k = J_i^{k+1} - J_i^k. \tag{4.1}$$

By definition,

$$\cos^2 \theta_{r^{k+1} J_i^{k+1}} = \frac{(r^{k+1}, J_i^{k+1})^2}{(r^{k+1}, r^{k+1})(J_i^{k+1}, J_i^{k+1})}. \tag{4.2}$$

After substituting $J_i^k + \Delta_i^k$ for J_i^{k+1} in (4.2), expanding the inner products, and simplifying [22, pp. 37–41] shows:

$$\cos^2 \theta_{r^{k+1} J_i^{k+1}} = \cos^2 \theta_{r^{k+1} \Delta_i^k}[\varphi_i^k] \tag{4.3}$$

where

$$\varphi_i^k = \left[1 - \frac{Z_i^k}{(\Delta_i^k, \Delta_i^k) + Z_i^k} + Q_i^k\right]$$

$$Z_i^k = [2(J_i^k, \Delta_i^k) + (J_i^k, J_i^k)]$$

$$Q_i^k = \frac{(\Delta_i^k, \Delta_i^k)}{(r^{k+1}, \Delta_i^k)^2}\left[\frac{(r^{k+1}, J_i^k)(J_i^k, J_i^k + 2\Delta_i^k)}{(\Delta_i^k, \Delta_i^k) + (J_i^k + 2\Delta_i^k)}\right]$$

In the event that column i was contained in the projection subspace used to obtain x^{k+1}, then

$$\cos^2 \theta_{r^{k+1} J_i^{k+1}} = \cos^2 \theta_{r^{k+1} \Delta_i^k}\left[1 - \frac{Z_i^k}{(\Delta_i^k, \Delta_i^k) + Z_i^k}\right].$$

The φ algorithm specified below chooses a projection subspace on the basis of the magnitude of the values of φ_i^k, $i = 1, 2, \ldots, n$. That is, column i is included in B_k whenever

$$(\Delta_i^k, \Delta_i^k) > (J_i^k, J_i^k) V$$

for some tolerance V. (J_i^k, J_i^k) is used rather than Z_i^k because (1) its value tends to dominate Z_i^k and (2) it requires less computation.

φ Algorithm

P1. Select a tolerance V
P2. Set $B_k \leftarrow \phi$
P3. $i \leftarrow 1$
P4. Compute (Δ_i^k, Δ_i^k)
P5. Compute (J_i^k, J_i^k)
P6. Compare (Δ_i^k, Δ_i^k) to $V^*(J_i^k, J_i^k)$
If $(\Delta_i^k, \Delta_i^k) > V^*(J_i^k, J_i^k)$, $B_k \leftarrow B_k \cup \{i\}$
P7. $i \leftarrow i + 1$
P8. Repeat P4, P5, P6, and P7 until $i > n$
P9. If $B_k = \phi$ set $B_k = \{1, 2, \ldots, n\}$

Values of V in the interval (0.05, 0.15) have been found to give the best results when the φ algorithm is used to select projection subspaces for the projection method.

V. OPERATION COUNTS

The amount of computation required at one step of the projection method, using either the φ algorithm or angle algorithm depends on n_k.

The number of multiplications required to determine B_k using the angle algorithm is $2n^2 + n$. To determine B_k using the φ algorithm require $2n^2$ multiplications.

The total multiplication counts required to determine B_k using the angle algorithm, create the n_k dimensional linear system, and then to solve the linear system using Gaussian elimination is:

$$2n^2 + n + 1/2[n_k(n_k - 1)n] + n_k^3/3 + n_k^2 \tag{5.1}$$

If $n_k = n$, then (5.1) reduces to $5/6n^3 + 5/2n^2 + n$.

The total multiplication counts required by the projection method for one iteration using the φ algorithm is:

$$2n^2 + 1/2[(n_k)(n_k - 1)n] + n_k^3/3 + n_k^2. \tag{5.2}$$

If $n_k = n$, the (5.2) reduces to $5/6n^3 + 7/2n^2$.

Using the above expressions for multiplication counts, it can be seen, for example, when $n = 10$, that the use of the projection method with either algorithm and $n_k > 4$ will require more multiplications than Newton's method. This extra computation can potentially be offset by larger reductions in the norm of the residue vector which might be obtained or by requiring fewer total iteration steps to reach a given accuracy. The examples in the next section illustrate this point.

VI. EXPERIMENTAL RESULTS

The numerical experiments comparing the angle algorithm and φ algorithm used in conjunction with the projection method to Newton's method were carried out in FORTRAN on a CDC 6400. In order to measure just the effect of the subspace selection algorithms, no other computational accelerations were applied to either the projection method or to Newton's method. Exact Jacobians were computed at each step for both Newton's method and the projection method. In addition, the change components were computed using a Gaussian elimination routine with full pivoting.

The comparison between methods is based upon the number of iteration steps required for solution and the central processing time required to reduce the norm of the residue vector to a value less than 1×10^{-5}. Since Newton's method and the n-dimensional projection method are mathematically equivalent, either method could be used to obtain the change components for the projection method at an iteration step when $B_k = \{1, 2, \ldots, n\}$. Consequently, Newton's method is compared to the projection method which uses Newton's method for n-dimensional projections and the projection method which uses the normalized equation (2.1) for n-dimensional projections. In the event that a null subspace was chosen by either subspace selection algorithm, a subspace composed of all the columns of the Jacobian was used.

The test problems were chosen because they have been previously used for the purpose of comparing methods for solving nonlinear systems of equations. Examples 2 and 4 appeared in papers by Broyden[8, 11]. Example 3 is a two dimensional system of transcendental equations that were used by Brown[3] in testing his method.

Example 2

$f_1 = -(3+\alpha x_1)x_1 + 2x_2 - B$

$f_i = x_{i-1} - (3+\alpha x_i)x_i + 2x_{i+1} - B, \quad i = 2, 3, \ldots, n-1$

$f_n = x_{n-1} - (3+\alpha x_n)x_n - B$

$n = 10 \qquad x^0 = (-1, 1, -1, 1, -1, 1, -1, 1, -1, 1)$

$\alpha = 0.5$

$B = 1.0 \quad \|r^0\| = 18.17966$

Table 2. Results for solving example 2

Method	Algorithm	Tolerance	Steps	$\|r^f\|$	CPU Sec.
Newton	—	—	9	0.0000000	0.214
Projection*	Angle	0.25	8	0.0000000	0.139
Projection	Angle	0.25	8	0.0000000	0.181
Projection*	φ	0.1	6	0.0000051	0.158
Projection	φ	0.1	7	0.0000002	0.207

*n Dimensional projections were obtained using Newton's method.

The quantities shown in Table 1 above can be used to compare the projection method with either subspace selection algorithm to Newton's method. It is clear from the results contained in this table that as expected there is a computational advantage associated with using Newton's method for computing n-dimensional projections.

The subspaces chosen by the projection method in the solution of Example 2 are listed in Table 3.

Table 3. Subspaces chosen by the projection method in the solution of example 2

Step	Angle algorithm	φ Algorithm
1	(2, 4, 6, 8)	(2, 4, 6, 8)
2	(10)	(2, 4, 6, 8)
3	(7, 8)	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
4	(5, 10)	(10)
5	(2, 3)	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
6	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
7	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
8	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)	

Newton's method failed to converge to a solution for Example 3 below. The φ algorithm performed noticeably better than the angle algorithm on this problem. On Example 4, the angle algorithm performed better than the φ algorithm.

Example 3

$$f_1(x) = 1/2 \sin(x_1 x_2) - \frac{x_2}{4\pi} - \frac{x_1}{2}$$

$$f_2(x) = \left(1 - \frac{1}{4\pi}\right)(e^{2x_1} - e) + \frac{e x_2}{\pi} - 2e x_1$$

Table 4. Results for solving example 3 with: $x^0 = (1.0, 1.0)$, $r^0 = (-0.15884, -0.27222)$, $\|r^0\| = 0.31518$

Method	Algorithm	Tolerance	Steps	$\|r^t\|$	CPU Sec.
Newton	—	—	500	2.2778018	0.953
Projection*	Angle	0.25	117	0.0000083	0.245
Projection	Angle	0.25	117	0.0000082	0.254
Projection*	φ	0.1	25	0.0000050	0.069
Projection	φ	0.1	29	0.0000086	0.081

*n-Dimensional projections were obtained using Newton's method.

Example 4

$$f_i = (k_1 + k_2 x_i^2)x_i + 1.0 - k_3 \sum_{\substack{j=i-r_1 \\ j \neq i}}^{i+r_2} (x_j + x_j^2)$$

$$i = 1, 2, \ldots, n$$

$$n = 8$$

$$k_1 = k_2 = k_3 = 1$$

$$r_1 = r_2 = 1$$

Table 5. Results for solving example 4 with: $x^0 = (1, 1, 1, 1, 1, 1, 1, 1)$, $r^0 = (1, -1, -1, -1, -1, -1, -1, 1)$, $\|r^0\| = 2.82843$

Method	Algorithm	Tolerance	Steps	$\|r^t\|$	CPU Sec.
Newton	—	—	31	0.0000069	0.369
Projection*	Angle	0.25	31	0.0000069	0.461
Projection	Angle	0.25	30	0.0000014	0.441
Projection*	φ	0.1	22	0.0000000	0.253
Projection	φ	0.1	22	0.0000000	0.334

*n-Dimensional projections were obtained using Newton's method.

VII. DISCUSSION AND CONCLUSIONS

Newton's method can be accelerated by using the projection method in coordination with the subspace selection algorithms presented in this paper. It is clear from the examples in the previous section that there is no advantage computationally to using n-dimensional projections. However, it is also apparent that there is an advantage to using alternatives other than Newton's iteration at each step. The angle and φ algorithms dynamically determine projection alternatives to be used at each iteration.

Our current work includes the development of criteria for tolerance selection for both algorithms and also the application of other accelerations in coordination with the projection technique. In addition, we are attempting to distinguish subsets of nonlinear problems for which the subspace selection algorithms are particularly effective.

REFERENCES

1. W. F. Ames, *Nonlinear Partial Differential Equations in Engineering.* Academic Press, New York (1972).
2. P. T. Boggs, The solution of nonlinear systems of equations by A-stable integration techniques. *SIAM J. numer. Anal.* **8**, 767–785 (1971).
3. K. M. Brown, *A Quadratically Convergent Method for Solving Simultaneous Non-linear Equations.* University Microfilms, (1966).
4. K. M. Brown, Computer oriented algorithms for solving systems of simultaneous nonlinear algebraic equations. *Numerical Solution of Systems of Nonlinear Algebraic Equations* (edited by G. D. Byrne and C. A. Hall). Academic Press, New York (1973).
5. K. M. Brown and S. D. Conte, The solution of simultaneous nonlinear equations. *A.C.M. National Meeting Proceedings* **22**, 111–114 (1967).
6. K. M. Brown and J. E. Dennis, Jr., Derivative free analogues of the Levenberg-Marquardt and Gauss algorithms for nonlinear least squares approximation. *Numer Math.* **18**, 289–297 (1972).
7. K. M. Brown and W. B. Gearhart, Deflation techniques for the calculation of further solutions of a nonlinear system. *Numer. Math.* **16**, 334–342 (1971).
8. C. G. Broyden, A class of methods for solving nonlinear simultaneous equations. *Math. Comp.* **19**, 577–593 (1965).
9. C. G. Broyden, A new method of solving nonlinear simultaneous equations. *Comput. J.* **12**, 94–99 (1969).

10. C. G. Broyden, The convergence of single-rank quasi-Newton methods. *Math. Comp.* **24**, 365–382 (1970).
11. C. G. Broyden, The convergence of an algorithm for solving sparse nonlinear systems. *Math. Comp.* **25**, 285–294 (1971).
12. C. G. Broyden, Quasi-Newton or modification methods. *Numerical Solution of Systems of Nonlinear Algebraic Equations* (edited by G. D. Byrne and C. A. Hall). Academic Press, New York (1973).
13. D. F. Davidenko, On a new method of numerical solution of systems of nonlinear equations. *Dokl. Akad. Nauk SSSR* **88**, 601–602 (1953).
14. W. C. Davidon, Variable metric method for minimization. *Argonne National Laboratory Report* 5990 (1959).
15. W. C. Davidon, Variance algorithm for minimization. *Comput. J.* **10**, 406–410 (1968).
16. J. E. Dennis, Jr., On the convergence of Broyden's method for nonlinear systems of equations. *Math. Comp.* **25**, 559–567 (1971).
17. J. E. Dennis, Jr., Some computational techniques for the nonlinear least squares problem. *Numerical Solution of Systems of Nonlinear Algebraic Equations* (edited by G. D. Byrne and C. A. Hall). Academic Press, New York (1973).
18. J. E. Dennis, Jr. and J. J. More, A characterization of superlinear convergence and its application to quasi-Newton methods. *Math. Comp.* **28**, 549–560 (1974).
19. R. Fletcher and M. J. D. Powell, A rapidly convergent descent method for minimization. *Comput. J.* **6**, 163–168 (1963/1964).
20. L. Fox, *An Introduction to Numerical Linear Algebra.* Oxford University Press, New York (1965).
21. F. Freudenstein and B. Roth, Numerical solutions of systems of nonlinear equations. *J. ACM* **10**, 550–556 (1963).
22. D. D. Georg, Determining nonlinear projection subspaces. *National Technical Information Service (USERDA) Report No.* IS-3545 (1975).
23. D. D. Georg and R. F. Keller, A projection algorithm for solving nonlinear systems of equations. Unpublished paper presented at the *Mathematical Software Conference* II at Purdue University, May 1974, NTIS USERDA Report No. IS-M-16.
24. H. Hartley, The modified Gauss-Newton method for the fitting of nonlinear regression functions of least squares. *Technometrics* **3**, 269–280 (1961).
25. A. S. Householder, *The Numerical Treatment of a Single Nonlinear Equation.* McGraw-Hill, New York (1970).
26. D. G. Luenberger, The conjugate residual method for constrained minimization problems. *SIAM J. numer. Anal.* **7**, 390–398 (1970).
27. O. G. Mancino, Resolution by iteration of some nonlinear systems. *J. ACM* **14**, 171–190 (1967).
28. G. H. Meyer, On solving nonlinear equations with a one-parameter operator imbedding. *SIAM. J. numer. Anal.* **5**, 739–752 (1968).
29. J. J. More, Coercivity conditions in nonlinear complementary problems. *SIAM Review*, **15**, 1–16 (1974).
30. L. V. Nguyen and R. F. Keller, Projection-based methods for solving systems of n nonlinear equations in n unknowns. *National Technical Information Service (USERDA). Report No. IS-*3546, (1975).
31. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York (1970).
32. J. M. Ortega and W. C. Rheinboldt, A general convergence result for unconstrained minimization methods. *SIAM J. numer. Anal.* **9**, 40–43 (1972).
33. A. M. Ostrowski, *Solutions of Equations and Systems of Equations.* Academic Press, New York (1966).
34. M. J. D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* **7**, 155–162 (1964).
35. M. J. D. Powell, A method for minimizing a sum of squares of non-linear functions without calculating derivatives. *Comput. J.* **7**, 303–307 (1965).
36. L. B. Rall, *Computational Solution of Nonlinear Operator Equations.* Wiley, New York (1969).
37. S. Schechter, Iteration methods for nonlinear problems. *Trans. Am. math. Soc.* **104**, 179–189 (1962).
38. S. Schechter, Relaxation methods for convex problems. *SIAM J. numer. Anal.* **5**, 601–612 (1968).
39. L. K. Schubert, Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian. *Math. Comp.* **25**, 27–30 (1970).
40. D. F. Shanno, Conditioning of quasi-Newton methods for function minimization. *Math. Comp.* **24**, 647–656 (1970).
41. J. F. Traub, *Iterative Methods for the Solution of Equations.* Prentice-Hall, Englewood Cliffs, N. J. (1964).
42. R. S. Varga, *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, N.J. (1962).
43. F. J. Zeleznik, Quasi-Newton methods for nonlinear equations. *J. ACM* **15**, 265–271 (1968).